Incorporating Temporal Commonsense in a Conversational Chatbot

Project Mentor: Dr Koninika Pal

Pranav Guruprasad Rao - 112101038
Rachit Sandeep Jain - 112101019

Temporal Common Sense

What is it?

Temporal common sense refers to the ability to understand and reason about time-related aspects of events, actions, or information in a contextually meaningful way.

Example- "***I am going on a vacation***" and "***I am going on a walk***"

Never Explicitly mentioned that vacation is for several days and walk is for few hours

There are five temporal properties:

1. Duration
2. Temporal Ordering
3. Typical Time
4. Frequency
5. Stationarity

Never mentioned explicitly

Challenges in Understanding Temporal Common Sense

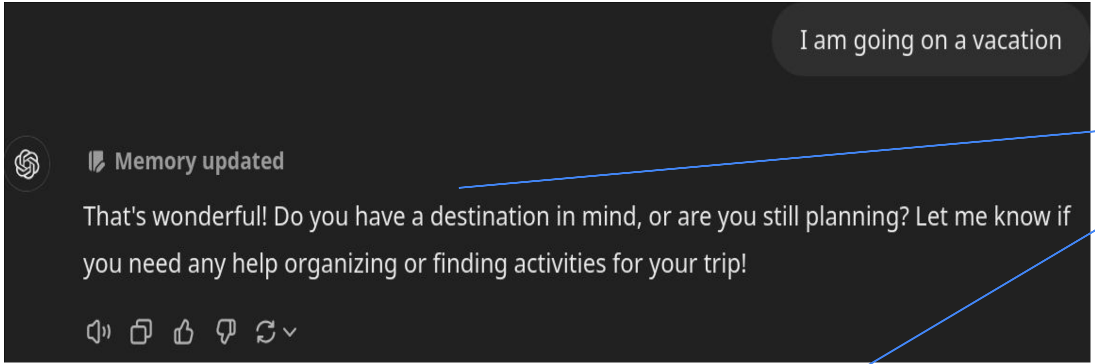
Patient - I would like to book an appointment with the doctor.
Receptionist - The doctor has currently gone on a vacation.

The Doctor is not available for couple of days.

What can the patient infer from this?

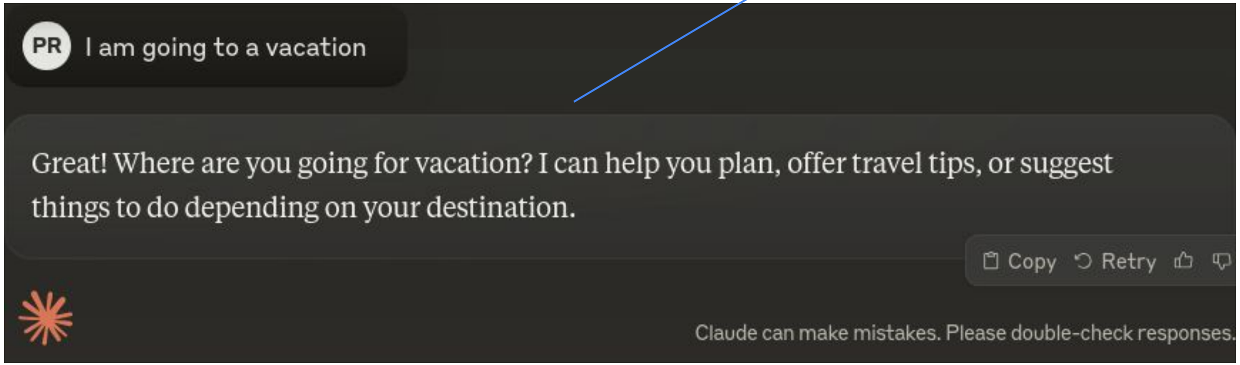
1. Never explicitly mentioned in text (hence known as ‘common sense’)
2. Thus difficult to extract it
3. Thus Language models are unable to capture it
4. Thus they are unable to respond in that direction
5. Even if they do, they do so infrequently

Challenges in Understanding Temporal Common Sense



Model sticks to this question for the following conversation!

Language Models unable to capture it and respond in temporal direction



What about "How long is your vacation?"

Problem Statement

1. Enhancing the context understanding for temporal common sense
2. Construct a temporal chatbot incorporating it
   a. Develop a logical schema to leverage or approximate temporal common sense

Where can this become critical?
Temporal Inferencing
Planning

Dataset

TIMEDIAL

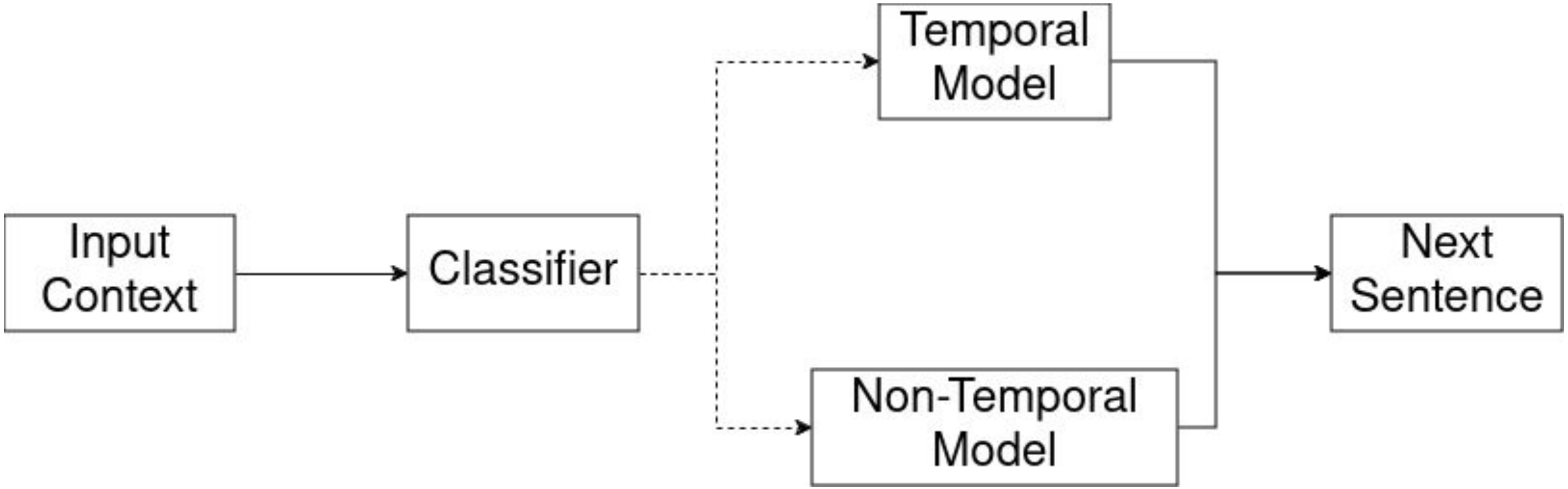
Training Dialogues	1395
Testing Dialogues	50
Training Data Size	27132
Testing Data Size	1268
Avg Rows per Dialogue	19

Table 3.1 Dataset Information

BIGBench
Temporal Ordering

Total Dialogues	7532
Training Dialogues	5649
Validation Dialogues	753
Testing Dialogues	1130

Proposed Architecture



Binary classifier

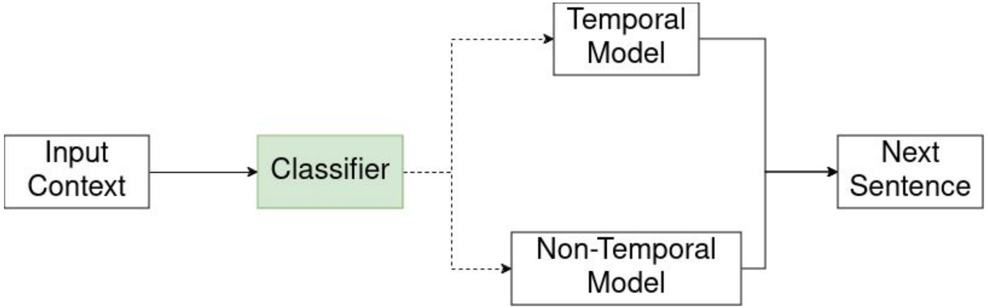
1. Given a context, identifies whether the next sentence requires temporal reasoning or not.

Dataset Preparation:

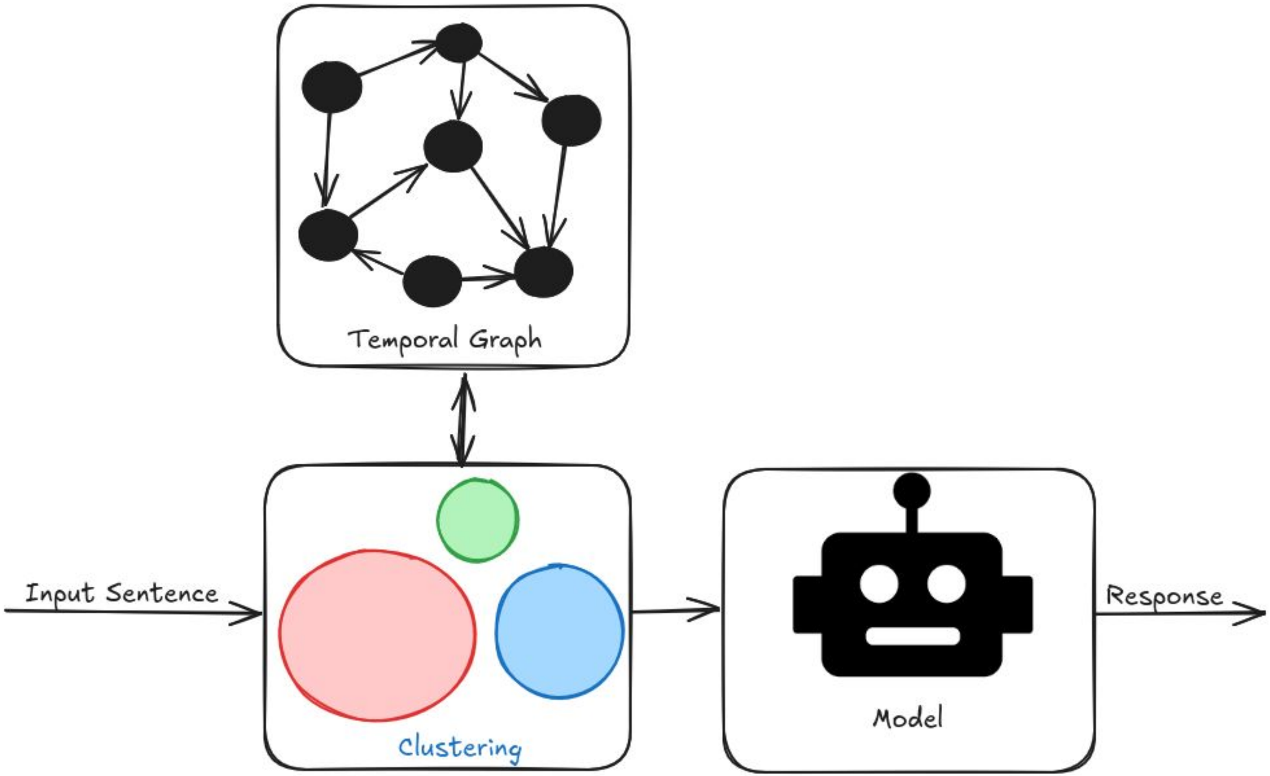
1. Each next sentence was converted into binary labels (0 or 1).

Temporal Label Generation:

1. spaCy
2. Custom rules
   a. Temporal adverbs
   b. Prepositions related to time
   c. Common time markers
   d. Auxiliary verbs



Temporal Pipeline



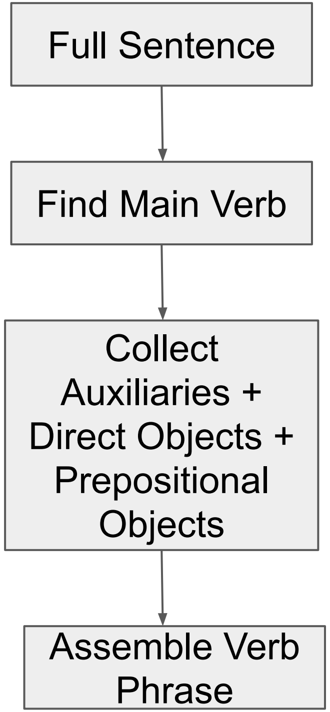
Event Extraction

1. Used spaCy in identifying key events or actions described in a sentence.

Example:

She has submitted the report on Monday.

Verb Phrase: has submitted report Monday



Time Extraction

1. spaCy
2. HeidelTime
3. Custom rules -
    a. Temporal adverbs like when, how often
    b. Preposition of time like during, after
    c. Common markers like yesterday and tomorrow

Encoder based Prediction

We used a BERT-based Encoder in order to associate events with their temporal values.
Training method - MLM - Masked Language Modeling on graph nodes

Difficult to scale as the graph increases in size
Increases noise in fetching data from the graph - opted for accuracy over speed

Clustering

We create clusters based on the verbs extracted.

Example of a Cluster

Temporal Graph

Aim - Event Linking
Edge exists between two events occur consecutively or are semantically linked

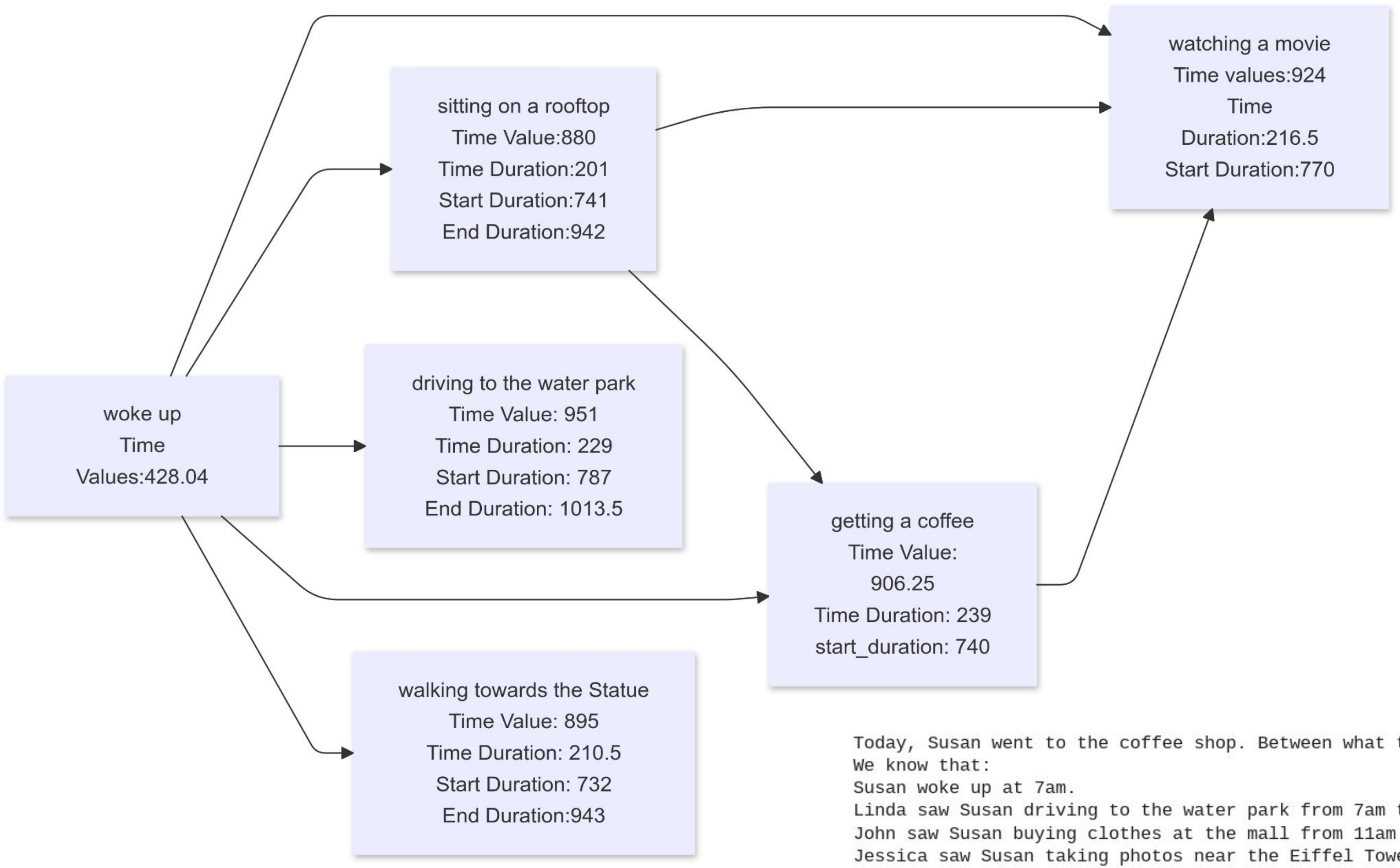
Node Attributes

1. Event: Event name
2. Temporal Range: A List of various time points and time intervals in which this event was observed in the conversations
3. Phrase: Different Verb Phrases related to the Event
4. Embeddings: Embeddings of the phrases
5. Average Embedding: Average embeddings of the Embeddings
6. Metadata: to store any additional metadata

Insertion of a new Node

When a new event arrives, 4 points to see:

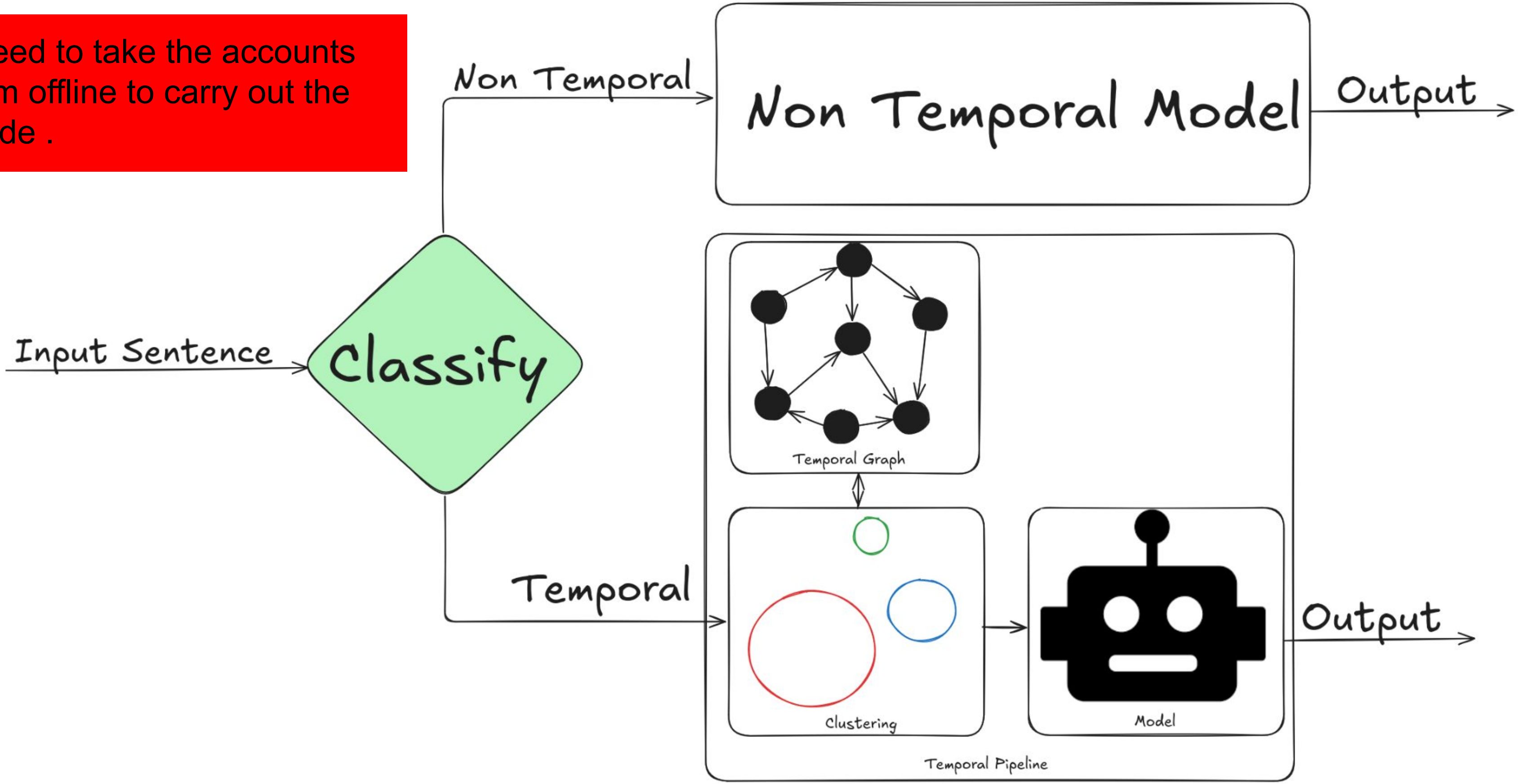
1. If the event **does not exist** → Create a **new GraphNode.**
2. Update the node by adding **temporal points, phrases,** and **metadata.**
3. If the new **phrase** is **similar** to an existing node → **merge** it.
4. If the phrase is **too different** → **create a sibling node** for the same event.




```
Today, Susan went to the coffee shop. Between what times could they have gone?
We know that:
Susan woke up at 7am.
Linda saw Susan driving to the water park from 7am to 11am.
John saw Susan buying clothes at the mall from 11am to 12pm.
Jessica saw Susan taking photos near the Eiffel Tower from 12pm to 1pm.
Steven saw Susan buying lunch at the deli from 1pm to 2pm.
Thomas saw Susan reading at the library from 2pm to 6pm.
The coffee shop was closed after 9pm.
Between what times could Susan have gone to the coffee shop?
```

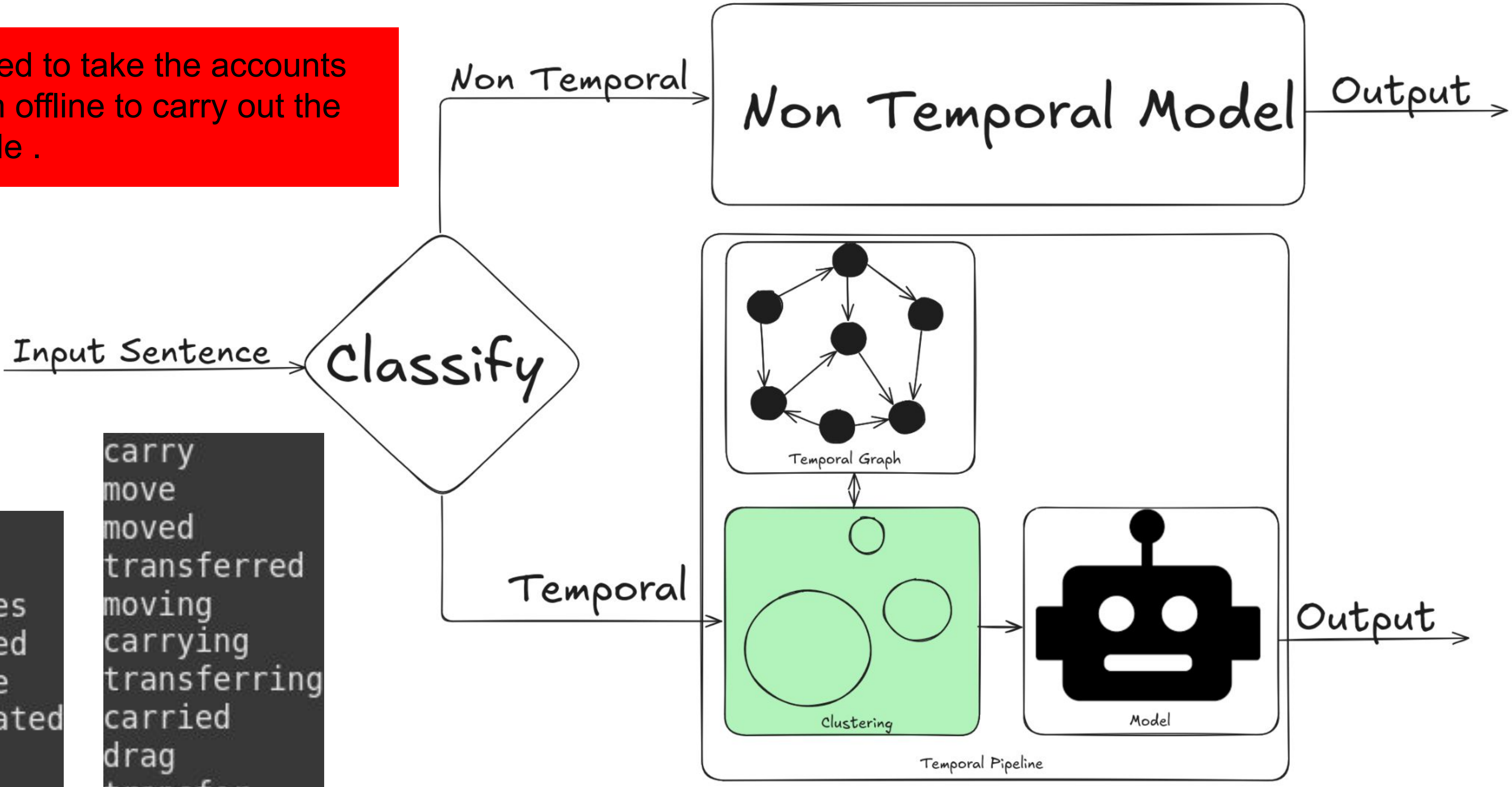
Overview of Entire Process

We need to take the accounts system offline to carry out the upgrade .



Overview of Entire Process

We need to take the accounts system offline to carry out the upgrade .

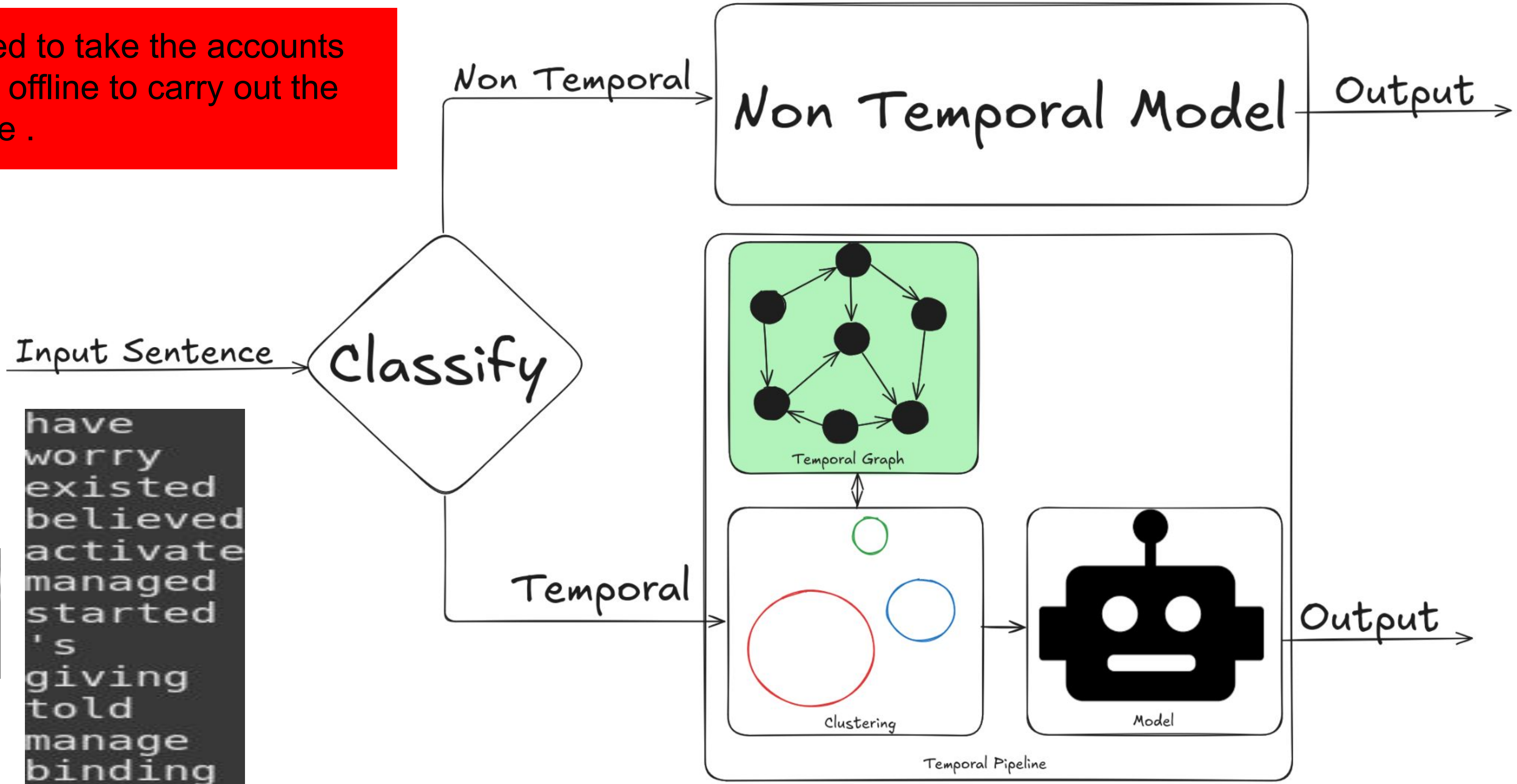



```
need
needs
requires
required
require
stipulated
needed
needing
```

```
carry
move
moved
transferred
moving
carrying
transferring
carried
drag
transfer
```

Overview of Entire Process

We need to take the accounts system offline to carry out the upgrade .

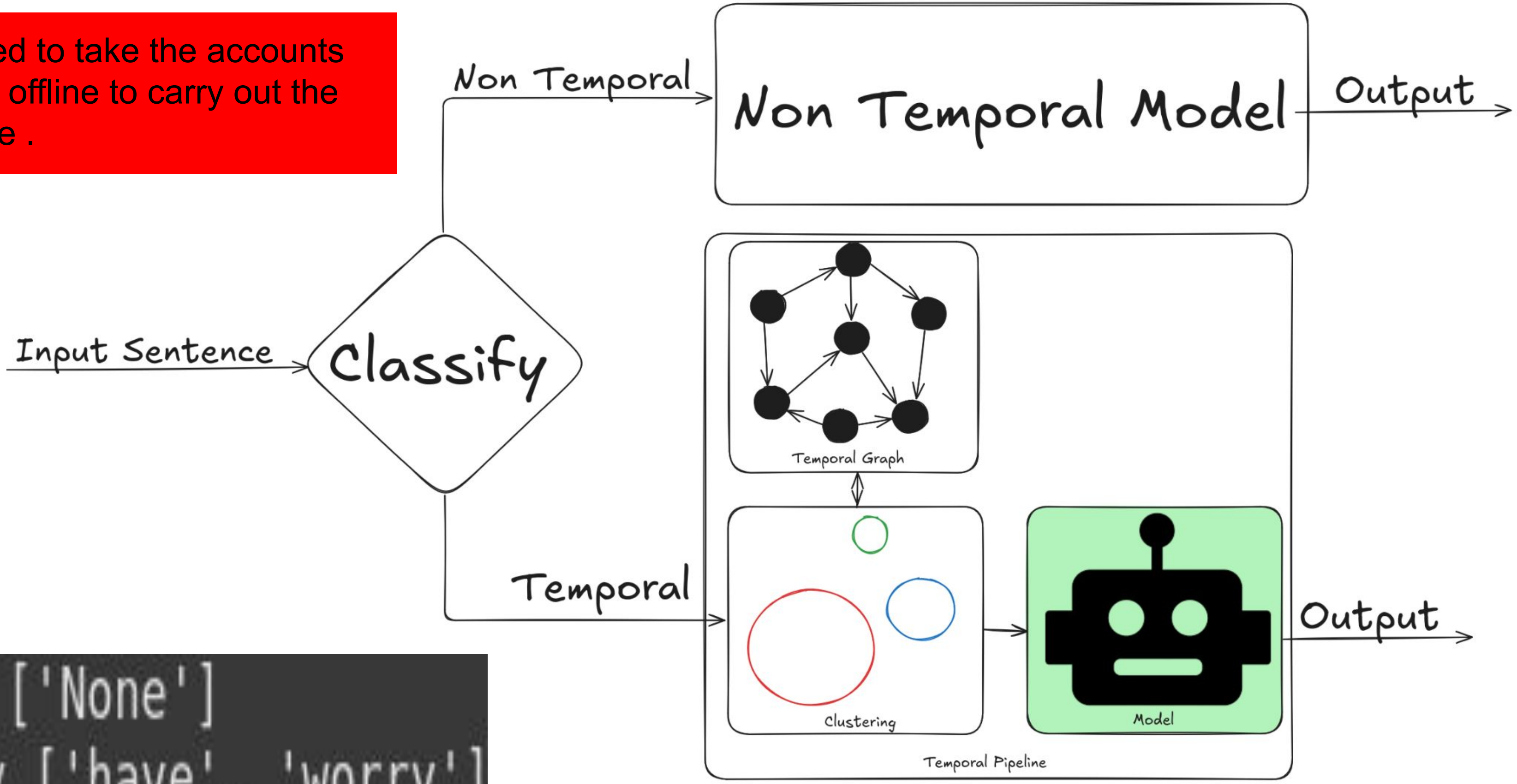


have
is

have
worry
existed
believed
activate
managed
started
's
giving
told
manage
binding
fill

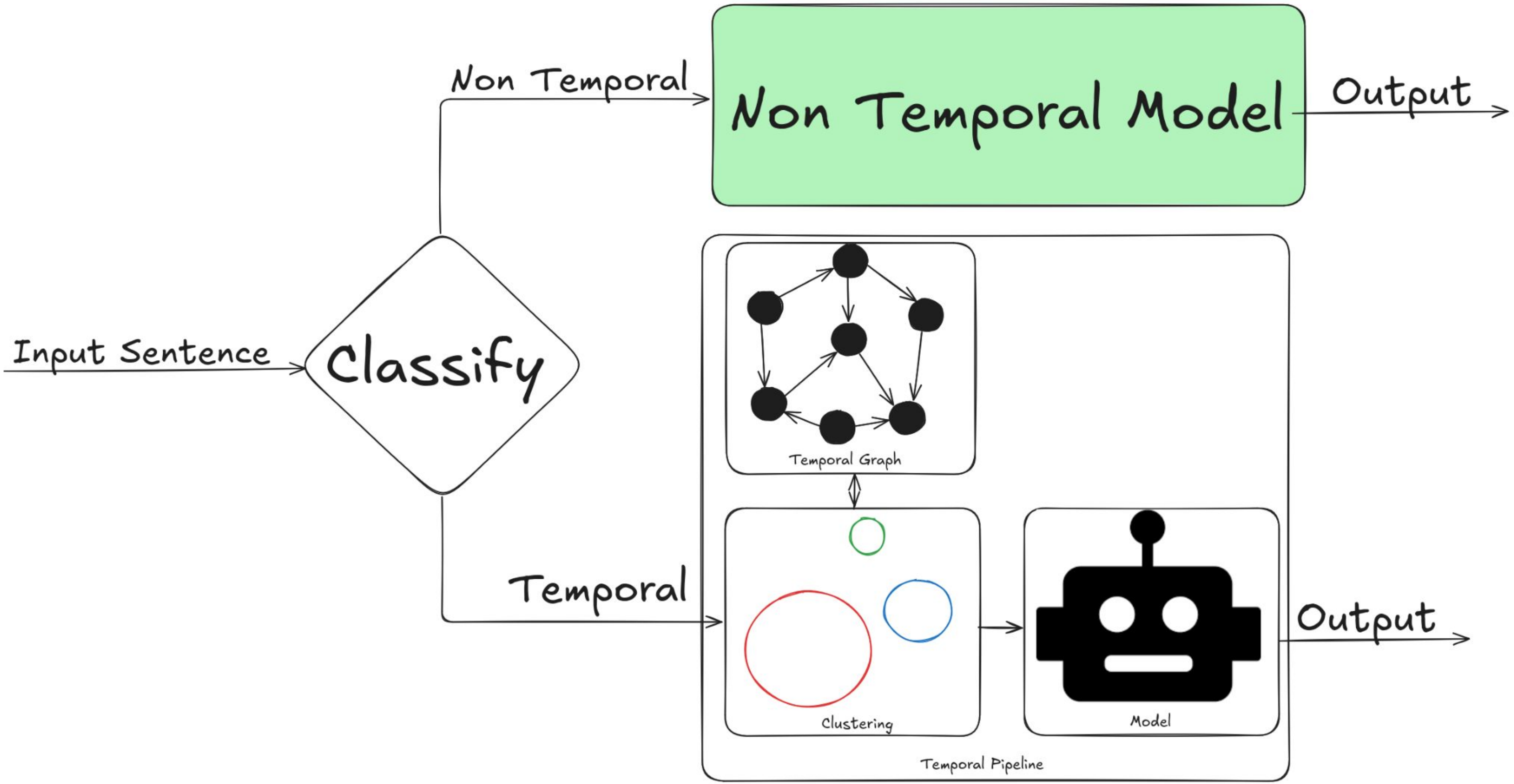
Overview of Entire Process

We need to take the accounts system offline to carry out the upgrade .




```
need ['None']
carry ['have', 'worry']
```

Overview of Entire Process



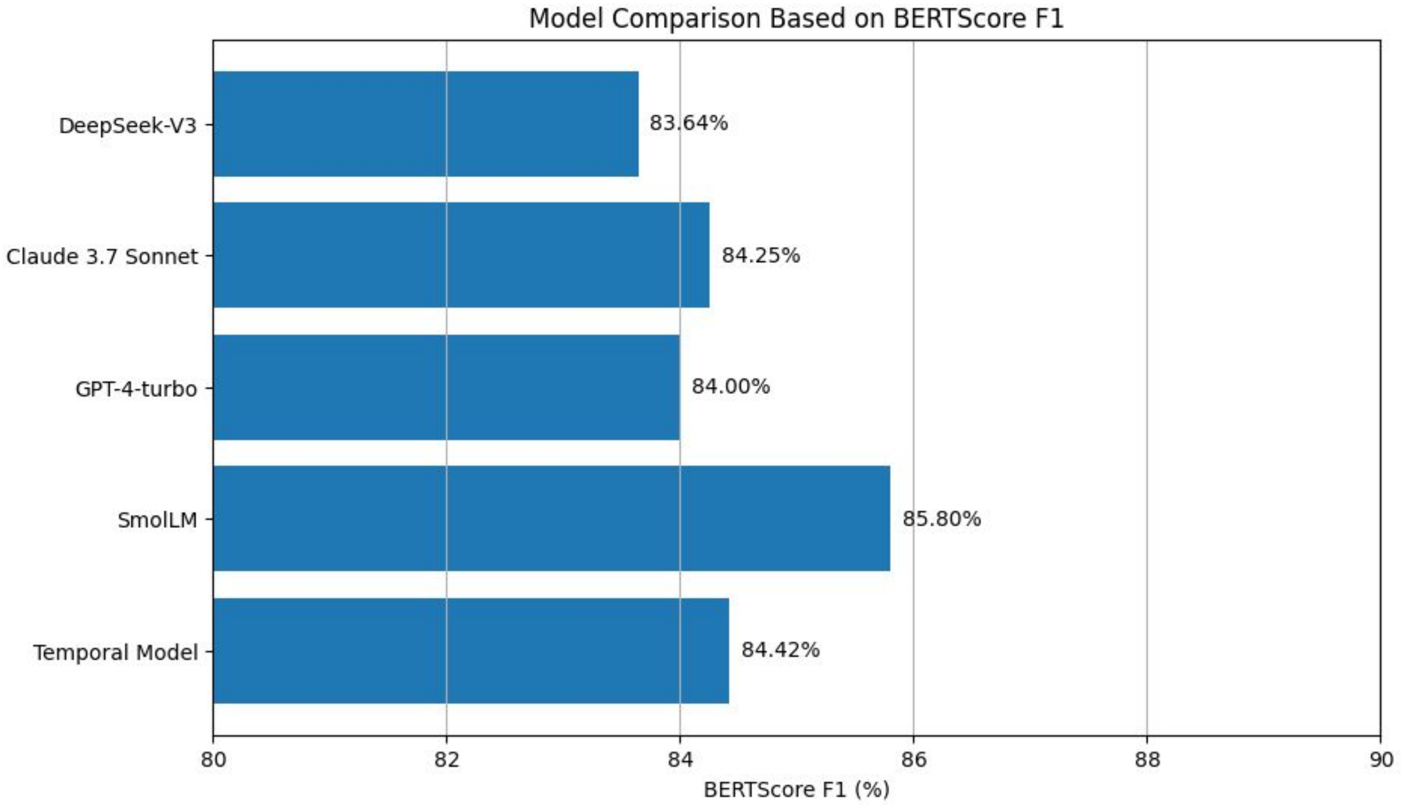
Pipeline in a Nutshell

1. Classify as Temporal or Non-temporal
2. If Temporal:
    a. Get the verbs and phrases in the sentence
    b. Find the clusters for the verbs
    c. Get the nodes for each clustered verbs
    d. Find the node most similar to the given phrase
    e. Get at most 5 of its popular neighbors (based on their number of phrases) to cue the model
    f. Temporal Model responds
3. Else:
    a. Non temporal responds

Training Process

- Model: Flan-T5-Base
- Data: Custom dialogue pairs (Input → Output)
- Optimizer: AdamW (Learning rate=3e-4, Weight Decay=0.01)
- Batch Size: 4 (train) / 2 (eval)
- Epochs: 10
- Dropout: 0.5
- Trainer: Huggingface Seq2SeqTrainer
- Metrics: ROUGE (train), BERTScore (eval)
- Checkpoints: Auto every epoch + manual every 5 epochs

Results



Model Comparison Based on BERTScore F1
DeepSeek-V3
83.64%
Claude 3.7 Sonnet
84.25%
GPT-4-turbo
84.00%
SmolLM
85.80%
Temporal Model
84.42%
80
82
84
86
88
90
BERTScore F1 (%)

Thank You